



An Analytic Expression of the Key Rank and a Saddle-Point Approximation

Mathieu des Noes 

Univ. Grenoble Alpes, CEA, LETI, MINATEC Campus, Grenoble, France

Abstract. Key ranking is a critical subject for evaluators since it measures the remaining complexity after a side-channel attack without the need to implement the key enumeration algorithm. This paper proposes a novel method based on the knapsack counting problem. While the existing literature proposes a solution of this problem with an algorithm originating from linear programming, we propose a new computation inspired by a recent work in statistical physics. The partition function of the knapsack problem, which is equivalent to the key rank, is derived with an analytic expression. This is used to prove the mathematical equivalence between the knapsack and histogram-based methods. In addition, a saddle-point approximation of the key rank is computed from the analytic expression. A very simple mathematical formula is derived. Simulation results show that the approximation is very tight. In addition, the execution time of the approximation is fast and linear in the key size, making it suitable for scaling to very large keys.

Keywords: Side-Channel · Key Rank · Complexity · Saddle-Point

1 Introduction

Side-channel attacks (SCAs) aim to recover the secret key used by a device when running a cryptographic algorithm (e.g., AES or RSA). In most cases, these algorithms split the key into subkeys to allow for efficient processing. SCA uses a divide-and-conquer approach to attack each subkey individually. In practice, it produces a vector of scores, and the true subkey is expected to be the one with the lowest score. In the end, the adversary gets a matrix of scores where each column is ranked in ascending order. In a favorable case, all subkeys are ranked first and the attack is successful. However, this situation does not occur systematically. The attacker must therefore implement a key search that works by ranking all keys according to their scores. This is done using key enumeration algorithms. They require a lot of memory. From the point of view of the evaluator, it is sufficient to estimate the number of keys that are ranked lower than the true key. It corresponds to the number of keys that will be tested first by the enumeration algorithm. This estimation is done by key ranking algorithms. It is a critical issue for evaluators because it measures the remaining complexity after SCA without the need to implement key enumeration. The most commonly used key ranking estimation techniques are usually companions to a key enumeration algorithm [VGRS13] [GGP⁺15] [MOOS15]. The goal is therefore to find an accurate, low complexity and low memory key ranking algorithm. A first step in this direction was made by Choudary and Popescu [CP17], who proposed to evaluate the security of the implementation with the Massey's guessing entropy (GM). They found a mathematical formula to bound the GM. Unfortunately, the GM is only related to the *average* key rank, while the evaluator is interested in the *actual* key rank.

E-mail: mathieu.desnoes@cea.fr (Mathieu des Noes)



In this paper, we propose a new estimation method that also provides a simple mathematical formula. The novelty is that it approximates the *actual* key rank and not the *average* value. This method is derived from a recent work in statistical physics. The key ranking algorithm based on the knapsack (KP) counting problem [MOOS15] serves as a starting point. While in [MOOS15] the authors propose to solve it with an algorithm originating from linear programming, here we propose an approximation inspired by statistical physics [Wil24]. The rank is equal to the partition function of the counting KP from a statistical physics point of view. The method detailed in [Wil24] is then adapted to our context in order to derive an analytic expression of the key rank. This formula is used to show the mathematical equivalence of the KP [MOOS15] and histogram [GGP⁺15] based algorithms. A saddle-point approximation of this formula is then used to evaluate the key rank. This results in a simple mathematical formula that is found to be very accurate compared to existing algorithms. **Paper Organization** Section 2 provides a state-of-the-art review of key rank estimation algorithms and motivates this work. Section 3 defines key rank estimation as a knapsack counting problem and derives an analytic expression. This result is used to establish the mathematical equivalence with the histogram method in a different way from [MMO18] and also the equivalence with the PRO algorithm described in [BLv15]. In Section 4, a saddle-point approximation of the key rank is detailed. This new estimation method is then compared to the key rank estimates obtained with the original knapsack counting algorithm [MOOS15] and the 'batch' histogram method [Gro18].

2 State of the art on key rank estimation algorithms

Side-channel attacks are not successful with probability 1. Therefore, key ranking algorithms aim to find the number of guesses an evaluator has to make to retrieve a secret key. This measures the remaining complexity after an attack. The side-channel attack produces a matrix of probabilities $p_{k,j}$ for the j^{th} hypothesis of the k^{th} subkey. The matrix has K rows and N columns, K is the number of subkeys (e.g. 16 for AES-128) and N is the number of hypotheses (e.g. 256 for a byte). Assuming that the probability distributions of the subkeys are independent random variables, the probability of a key $s = (s_0, \dots, s_{K-1})$ is $P_s = \prod_{i=0}^{K-1} P_i(s_i)$, where $P_i(s_i)$ is the probability that the i^{th} subkey is equal to s_i .

Definition 1 (Key rank). The key rank R is the number of key candidates that have a higher probability of being the correct key s^* :

$$R = \mathbf{card} \left(\{s \text{ s.t. } P_s > P_{s^*}\} \right)$$

where $\mathbf{card}(A)$ is the number of elements contained in the set A .

Key ranking algorithms can be classified in three categories: deterministic, sampling-based and entropy-based.

The deterministic algorithms were published first [VGRS13] [GGP⁺15][BLv15] [MOOS15]. They attempt to efficiently combine the subkey probabilities to provide an accurate estimate of the key rank. The optimal algorithm [VGRS13] builds iteratively the set of key candidates having a higher probability than the correct key (V1). The rank of the key is given by the number of elements of V1. The memory requirements of this algorithm are huge, making it impractical. Suboptimal key ranking algorithms have been proposed. They map the probabilities to integers in order to derive efficient counting algorithms. In [GGP⁺15], the authors propose to approximate the convolution of the discrete subkey log-probabilities by the convolution of histograms. Upper and lower bounds on the key rank are also derived. The memory consumption of this algorithm does not grow linearly with the number of subkeys. It is efficient for small to medium key sizes (e.g. AES), but its memory requirements increase too much for large key sizes (e.g., RSA). An adaptation of this algorithm for large key sizes has been proposed in [Gro18]. It reduces the complexity of

the histogram method at the expense of its accuracy (for key ranking). The PRO algorithm detailed in [BLv15] was found equivalent to the convolution of histograms [PGS15].

In [MOOS15], key ranking is formalized as a KP counting problem and an algorithm from linear programming is proposed to solve it. A graph is generated and the key rank is given by counting the number of paths of this acyclic graph. The depth of the graph is $N \times K$ long and its width is equal to W , the volume of the knapsack. Thus, the memory requirement of this algorithm does not grow linearly with the number of subkeys. In [MMO18], the authors show that their algorithm [MOOS15] is equivalent to the histogram algorithm [GGP⁺15] for a suitable choice of their respective discretization parameters (bin width and number of bits for the analog-to-digital conversion of the probabilities). These two algorithms require an optimized implementation to support large key sizes [YMO22]. In [DW22], the authors propose an algorithm denoted ESRank that approximates the probability distribution of the subkeys using exponential sampling techniques. Given a decreasing subkey probability distribution, the algorithm estimates the rank using the first b probabilities and then samples the distribution with an exponential increase in the chosen index. The exponential parameter $\gamma > 1$ allows to control the error ratio between the upper and lower bounds for any probability distribution of the subkeys. The practical experiment shows that ESRank achieves the same execution time as the histogram algorithm.

The sampling-based algorithms [ZDF20] [CDS23] approximate the key rank or Guessing Entropy (GE) by sampling from a subspace rather than the entire key space. The Monte-Carlo method is used in both cases. A uniform distribution is used in [ZDF20] for sampling the subkeys (GEEA), while a non-uniform sampling of the subkeys according to their score distribution is implemented in [CDS23](MCRank). For a similar accuracy, the execution time of MCRank is found to be one order of magnitude faster than the histogram algorithm. This algorithm also supports a variant for non-independent subkeys, e.g. for AES-256. It was however found in [HO24] that MCRank is not adapted for score-based distinguishing vectors output e.g. by a CPA.

The entropy-based algorithms [CP17] [ZDF20] [RPC22] do not measure the key rank, but aim to evaluate the security of the device by providing entropy-based information. In [CP17], the authors propose Massey's Guessing Entropy (GM) as an ultra-fast security evaluation tool. It is indeed ultra fast and scalable because it relies on mathematically derived bounds on the GM. In [RPC22] it was shown that the GM provides a lower bound on the average GE. In [ZDF20], the average GE is estimated based on a multivariate Gaussian distribution of the difference of the ranking score vectors and the pairwise success probabilities. However, an evaluator wants to estimate the exact key rank and its distribution, not its mean. In [MMOS16] the authors show that the arithmetic mean of the key rank is close to the worst case and not representative of the key rank distribution. Consequently, GM and average GE are not practical metrics for an evaluator.

Most of the key rank estimation algorithms presented above have a major drawback: the complexity or the execution time increase significantly with the key size. Scalability is therefore an important issue. This is also the case for GEEA when considering the accurate estimation of the parameters of the multivariate Gaussian model [YMO22]. The only highly scalable and ultra-low complexity method is GM estimation [CP17]. Unfortunately, it does not achieve our goal of exact key rank estimation. In this article, we propose a mathematically derived approximation of the exact key rank whose complexity grows linearly with key size.

The links between combinatorial optimization and statistical physics are well established [MM09]. Many approximation algorithms have been derived to count solutions to statistical mechanics problems.

Counting the number of solutions of the KP is equivalent to evaluating the partition function of the corresponding statistical physics system. In [Wil24], an approximation of the partition function for a large volume of the knapsack is computed and used to derive the

already known linear programming algorithms. We reuse the same methodology developed by the author and adapt it to the specificity of our multiple choice KP. An approximation to the exact key rank is provided with the advantage of a low-cost implementation of a mathematical expression. The advantages and limitations of the main key ranking estimation methods are summarized in Table 1. The user can choose a method based on its most important criteria (e.g., speed or existence of an error bound).

Table 1: Comparison of key rank estimation methods.

Method	Advantages	Limitations
Histogram [GGP ⁺ 15]	- Theoretical error bound - Fast for small to medium key sizes	- Complexity increases not linearly with key size - Need an optimized implementation for large key sizes
KP [MOOS15]	- Theoretical error bound - Fast for small to medium key sizes	- Complexity increases not linearly with key size - Need an optimized implementation for large key sizes
PRO [BLv15]	- Theoretical error bound - Fast for small to medium key sizes	- Complexity increases not linearly with key size - Need an optimized implementation for large key sizes
Histogram (batch) [Gro18]	- Theoretical error bound - Fast for large key sizes	- Complexity increases not linearly with key size - Need an optimized implementation for large key sizes - Loose error bounds
MCRank [CDS23]	- Theoretical error bound - Scalable to very large key sizes	- Execution time depends on the randomly selected samples (with resampling)
Massey's guessing entropy [CP17]	- Ultra low complexity - Theoretical bounds - Scalable to very large key sizes - No need of an optimized implementation for large key sizes	Compute the average GE instead of the key rank
Our solution	- Ultra low complexity - Scalable to very large key sizes - No need of an optimized implementation for large key sizes	No theoretical error bound

3 The knapsack problem and its partition function

The Knapsack Problem (KP) is a well-known combinatorial optimization problem [SDK75]. [KPP+04]. We consider N objects labeled with index $j = 0; \dots; N-1$, where each object can be selected or excluded from a set. If selected, the object j has a value v_j and a weight w_j . The goal is to find the combination of objects that maximizes the total value while remaining under a given total weight W . A given collection of objects is defined as $x = (x_0; x_1; \dots; x_{N-1})$ where $x_j = 1$ or 0 for object j selected or not. Thus, the goal of solving the KP is to find x that maximizes $\sum_{j=0}^{N-1} v_j x_j$ under the constraint $\sum_{j=0}^{N-1} w_j x_j < W$.

3.1 Analytic expression of the key rank

In our context, we need to solve a multidimensional KP. The output of the side channel attack is a matrix of scores or probabilities. These scores/probabilities are quantized to integer weights $w_{k,j}$ according to the algorithm 1 and sorted in ascending order [MOOS15]. The index $k \in \llbracket 0, K-1 \rrbracket$ is associated with the number of subkeys K , and $j \in \llbracket 0, N-1 \rrbracket$ is one of the N^{th} hypotheses (e.g. $N = 256$ for a byte-value model). The volume of the knapsack W is defined by $W = \sum_{k=0}^{K-1} w_{k,s_k^*}$, where w_{k,s_k^*} is the weight of the k^{th} subkey of the correct key and s_k^* its integer value. The key ranking problem consists in counting the number of solutions R to the KP:

$$\sum_{k=0}^{K-1} \sum_{j=0}^{N-1} w_{k,j} x_{k,j} < W$$

$x_{k,j} \in \{0, 1\}$ is a binary variable that models the selection of the value j of the variable k . In our KP, only one value can be selected for each variable. In addition, since the weights $w_{k,j}$ are integers, the constraint " $< W$ " is transformed in " $\leq W - 1$ ".

The problem is thus restated as follows:

$$\begin{aligned} \sum_{k=0}^{K-1} \sum_{j=0}^{N-1} w_{k,j} x_{k,j} &\leq W - 1 \\ \sum_{j=0}^{N-1} x_{k,j} &= 1 \quad \forall k \end{aligned} \tag{1}$$

In [MOOS15], the number of solutions (i.e., the rank) is measured by constructing a directed acyclic graph and counting the number of paths. This article proposes a different approach, inspired by the work of Williams in statistical physics [Wil24]. The rank is also defined by the partition function of the KP:

$$R = \sum_x \Theta \left(W - 1 - \sum_{k=0}^{K-1} \sum_{j=0}^{N-1} w_{k,j} x_{k,j} \right) \prod_{k=0}^{K-1} \delta \left(\sum_{j=0}^{N-1} x_{k,j}, 1 \right) \tag{2}$$

where $\sum_x$ is a notation for the sum over all the possible values of the state vector $(x_{0,0}, x_{0,1}, \dots, x_{K-1,N-1})$. $\Theta(j)$ is the discrete Heaviside step function:

$$\Theta(j) = \begin{cases} 1 & \text{if } j \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and $\delta(j, m)$ is the Kronecker symbol:

$$\delta(j, m) = \begin{cases} 1 & \text{if } j = m \\ 0 & \text{otherwise} \end{cases}$$

The first term of the equation counts '1' if the combination of subkeys satisfies the KP. The second term ensures that only one value is selected for each subkey.

Theorem 1. *The key rank defined by Eq. 2 is also given by the following analytic expression:*

$$R = \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^W} \frac{1}{1-z} \prod_{k=0}^{K-1} \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right) \quad (3)$$

where Γ is a counterclockwise contour strictly contained in the unit circle.

Proof. The discrete Heaviside function can be written as a series:

$$\Theta(j) = \sum_{m=0}^{+\infty} \delta(j, m)$$

In addition, $\delta(j, m)$ is expressed as a contour integral:

$$\delta(j, m) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^{j-m+1}}$$

where Γ is a counterclockwise contour strictly contained in the unit circle. To check this formula, set $z = re^{i\theta}$ and integrate over the interval $[0, 2\pi]$. The result is zero unless $j = m$. Consequently, the discrete Heaviside function is also expressed as a contour integral:

$$\Theta(j) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^{j+1}(1-z)}$$

Inserting this result into Eq. 2, we obtain:

$$R = \sum_x \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^W} \frac{1}{1-z} z^{\sum_k \sum_j x_{k,j} w_{k,j}} \prod_{k=0}^{K-1} \delta \left(\sum_{j=0}^{N-1} x_{k,j}, 1 \right)$$

The summation $\sum_x$ is then split into K sums $\sum_{x_0} \sum_{x_1} \cdots \sum_{x_{K-1}}$:

$$R = \sum_{x_0} \cdots \sum_{x_{K-1}} \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^W} \frac{1}{1-z} \prod_{k=0}^{K-1} z^{\sum_j x_{k,j} w_{k,j}} \delta \left(\sum_{j=0}^{N-1} x_{k,j}, 1 \right)$$

where $x_k = (x_{k,0}, \dots, x_{k,N-1})$. The equation can be rewritten as follows:

$$R = \sum_{x_1} \cdots \sum_{x_{K-1}} \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^W} \frac{1}{1-z} \left(\prod_{k=1}^{K-1} z^{\sum_j x_{k,j} w_{k,j}} \delta \left(\sum_{j=0}^{N-1} x_{k,j}, 1 \right) \right) \left(\sum_{x_0} z^{\sum_j x_{0,j} w_{0,j}} \delta \left(\sum_{j=0}^{N-1} x_{0,j}, 1 \right) \right)$$

One notes that

$$\sum_{x_0} z^{\sum_j x_{0,j} w_{0,j}} \delta \left(\sum_{j=0}^{N-1} x_{0,j}, 1 \right) = \sum_{j=0}^{N-1} z^{w_{0,j}}$$

Consequently, the expression of R becomes:

$$R = \sum_{x_1} \cdots \sum_{x_{K-1}} \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^W} \frac{1}{1-z} \left(\sum_{j=0}^{N-1} z^{w_{0,j}} \right) \prod_{k=1}^{K-1} z^{\sum_j x_{k,j} w_{k,j}} \delta \left(\sum_{j=0}^{N-1} x_{k,j}, 1 \right)$$

This computation is then applied successively to the other x_k 's to give Eq. 3. $\square$

We will now derive a more applicable analytic expression of Eq. 3.

Theorem 2. Let us define the polynomial $A(z) = \prod_{k=0}^{K-1} \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right) = \sum_j a_j z^j$. The key rank defined by Eq. 3 is also given by the following analytic expression:

$$R = \sum_{i=0}^{W-1} a_i \quad (4)$$

Proof. Let us define the power series $G(z) = \sum_j g_j z^j$. The Cauchy coefficient is defined as follows:

$$g_j = \frac{1}{2\pi i} \oint_{\Gamma} \frac{G(z)}{z^{j+1}} dz$$

One recognizes in Eq. 3 the Cauchy's coefficient of a power series. Let us define the function $G(z) = \frac{1}{1-z} \prod_{k=0}^{K-1} \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right)$. The expression $\frac{1}{1-z}$ can also be written as a power series: $\frac{1}{1-z} = \sum_{j=0}^{\infty} z^j$ for $|z| < 1$. The function $G(z)$ is thus the product of the polynomial $A(z)$ with a power series and is itself a power series. The coefficients g_j are equal to the convolution of the a_j 's with the coefficients 1 of the power series $\sum_{j=0}^{\infty} z^j$: $g_j = \sum_{n=0}^j a_n$. The rank is finally given by its coefficient g_{W-1} : $R = g_{W-1}$. $\square$

Algorithm 1 Quantization of scores [MOOS15]

```

max_score = max(-log(p,:))
α = log2(max_score)
q = 2ν-α
for k = 0, 1, ..., K - 1 do
    min_scorek = min(-log(pk,:))
    q = (max_score - min_scorek) / 2ν                                ▷ Quantization step
    for j = 0, 1, ..., N - 1 do
        sk,j = ⌊ -log(pk,j) * q ⌋
    end for
    wk,:  = sort(sk,: )                                                ▷ Sort by increasing order
    wk,:  = wk,:  - wk,0                                                ▷ subtract the smallest value per key byte
end for

```

3.2 Equivalence with the histogram method

The key rank estimation based on the convolution of histograms is presented in [GGP⁺15]. The histograms are generated with Algorithm 2 and the rank is estimated with Algorithm 3. One can easily see a similarity between Algorithms 1 and 2 and a connection with Theorem 2.

The variable $s_{k,j}$ in Algorithm 1 is identical to *Hist_index* in Algorithm 2. Consequently, the value $H[k, w_{k,j}]$ represents the multiplicity of scores $-\log(p_{k,i})$ that are quantized to the same value $w_{k,j}$. Hence

$$A_k(z) = \sum_{j=0}^{N-1} z^{w_{k,j}} = \sum_{j=0}^{2^\nu-1} H[k][j] z^j$$

In addition W is also the index of the bin containing the key. The function $A_k(x) = \sum_{j=0}^{N-1} z^{w_{k,j}} = \sum_{j=0}^{2^\nu-1} H[k][j] z^j$ is thus a polynomial representation of the histogram $H[k]$. The coefficients of the polynomial $A_k(x)A_l(x)$ are given by the convolution of the coefficients of each polynomial and thus by the convolution of the histograms $H[k]$ and

$H[l]$. As a result, the computation of $A(z) = \prod_{k=0}^{K-1} \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right) = \sum_j a_j z^j$ implements the convolution of the K histograms. Finally, according to Theorem 2, the estimation of the key rank is $R = \sum_{i=0}^{W-1} a_i$ which is equivalent to $R_H = \sum_{i=0}^{W-1} H_{curr}[i]$ in Algorithm 2. Thus, the KP [MOOS15] and the histogram [GGP⁺15] based methods implement the same calculus in different ways. This equivalence was already established in [MMO18] with a different method.

Algorithm 2 Computation of histograms (2^ν bins)

```

 $H[0, \dots, K-1][0, \dots, N-1] = 0$  ▷ Initialization of the  $K$  histograms
for  $k = 0, 1, \dots, K-1$  do
     $min\_score = \min(-\log(p_{k,:}))$ 
     $max\_score = \max(-\log(p_{k,:}))$ 
     $q = (max\_score - min\_score) / 2^\nu$  ▷ Set bin width
    for  $j = 0, 1, \dots, N-1$  do
         $Hist\_index = \lfloor (-\log(p_{k,j}) - min\_score) / q \rfloor$  ▷ Index of the bin
         $H[k, Hist\_index] += 1$ 
    end for
end for

```

Algorithm 3 Rank estimation with histograms

```

Initialization:  $H_{curr} = H[0][0, \dots, N-1]$ 
Histogram convolution:
for  $k = 1, \dots, K-1$  do
     $H_{curr} = H_{curr} * H[k]$  ▷  $*$  is the convolution operator
end for
 $R_H = \sum_{i=0}^{W-1} H_{curr}[i]$ 

```

3.3 Equivalence with the PRO algorithm

We also note that Theorem 2 corresponds to a mathematical description of the PRO algorithm detailed in [BLv15]. The algorithm is first described with our notations and the the equivalence is established.

Let us first define the concept of multiset: it is a set that allows for multiple instances for each of its elements. We define K multisets $\mathcal{S}$, one for each subkey weight distributions $(w_{k,:})$ and $\mathcal{P}$ the multiset that select one element from each multiset $\mathcal{S}$. A *generalized polynomial* F is a function of the following form: $F(x) = \sum_{r \in \mathbb{R}} F(r) x^r$ such that $F(r) \neq 0$ for only finitely many $r \in \mathbb{R}$. The *distribution* of a generalized polynomial F is:

$$distr(F)(h) = \sum_{r < h} F(r)$$

With these definitions, the PRO algorithm is given in Algorithm 4. The generalized polynomial $G_{\mathcal{P}}(x)$ is equivalent to the term $\prod_{k=0}^{K-1} \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right)$ in Eq. 3. The generalized polynomial $G_{\mathcal{P}}(x)$ is also similar to the polynomial $A(z)$ defined in Theorem 2. Also, the computation of the distribution of $distr(G_{\mathcal{P}})(W)$ is equal to the rank computed by Eq. 4. Consequently, the PRO algorithm proposed in [BLv15] gives the same result as the KP and Histogram algorithms.

Algorithm 4 Polynomial Rank Outlining Algorithm (PRO)

Input: The weight matrix $w_{k,j}$ and the key weight W .

$$G_{\mathcal{P}}(x) = \prod_{S \in \mathcal{P}} \sum_{w \in S} x^w$$

$$R_{PRO} = \text{distr}(G_{\mathcal{P}})(W)$$

4 A saddle-point approximation of the key rank

Our goal now is to obtain an accurate approximation of Eq. 3. To do this, we use the method of saddle-point approximation. It is the complex analytic counterpart of the Laplace method for evaluating real integrals depending on a large parameter. A complete detailed description can be found in Chapter VIII of [FS09]. In the following, first the principle of the approximation is given and then the application to key rank is detailed.

4.1 Saddle-point method for Cauchy coefficient integrals

The definition of a saddle-point is first given. It will be used to derive the key rank approximation.

Definition 2 (Saddle-point [FS09]). Consider any function $f(z)$ analytic for $z \in \Omega$, where Ω is some domain of $\mathbb{C}$. A saddle-point is a point $z_s \in \Omega$ such that $f(z_s) \neq 0$ and $f'(z_s) = 0$.

Assume one wants to evaluate the following integral:

$$R(W) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{G(z)}{z^{W+1}} dz$$

where Γ is a counterclockwise contour strictly contained in the unit circle. It is defined as a circle $z = z_0 e^{i\theta}$ with $0 < z_0 < 1$ and $-\pi \leq \theta \leq \pi$. The saddle-point approximation assumes that the result of the integral over the contour Γ comes mostly from the neighborhood of the maximum of the function $f(z) = \log(G(z)) - W \log(z)$ at point z_s .

The function $f(z)$ can be decomposed into its real and imaginary parts: $f(z) = u(x, y) + iv(x, y)$. If W is large and positive, we expect the exponential $e^{f(z)}$ to be largest at the maximum of $u(x, y)$. Thus, if the contour passes through a point z_s where $u(x, y)$ is maximum, the contribution to the integral may come mostly from the neighborhood of z_s . On the other hand, if W is large, a small change in $v(x, y)$ will cause very fast oscillations in the imaginary part of $f(z)$. These oscillations make the contribution of the real part of $f(z_s)$ negligible. Therefore, we also want $v(x, y)$ to vary as slowly as possible near z_s . A necessary condition is that the derivative of $v(x, y)$ vanishes at z_s . This and the fact that $u(x, y)$ should be maximum at z_s leads to $f'(z_s) = 0$. This together with the Cauchy-Riemann conditions implies that $\partial u / \partial x = \partial u / \partial y = 0$ at z_s . It is therefore a saddle point of the surface $u(x, y)$, i.e. neither a local maximum nor a minimum. As an illustration, Figure 1 plots the surface of the real part of $f(z)$ near $z_s = 1$ with $f(z)$ defined by Eq. 6, which shows a clear saddle point.

Lemma 1 (Existence of a unique positive real saddle-point (Corollary VIII.1 of [FS09])). *Let $G(z)$, not a polynomial, be analytic at 0 with non-negative coefficients and radius of convergence $R_c \leq +\infty$. Then there exists a unique positive roots z_0 of $z_0 \frac{G'(z_0)}{G(z_0)} = W + 1$.*

In our case, $G(z)$ defined by Eq. 6 is not a polynomial, the coefficients g_j are positive (combination of positive coefficients $H[k][j]$) and the radius of convergence is $R_c = 1$. Consequently, there exists a unique real positive saddle-point $0 < z_0 < 1$ of $f(z)$.

The approximation described above is valid if the function $|f(z_0 e^{i\theta})|$ decays rapidly as θ increases. We will now give more details about the conditions needed to get an accurate

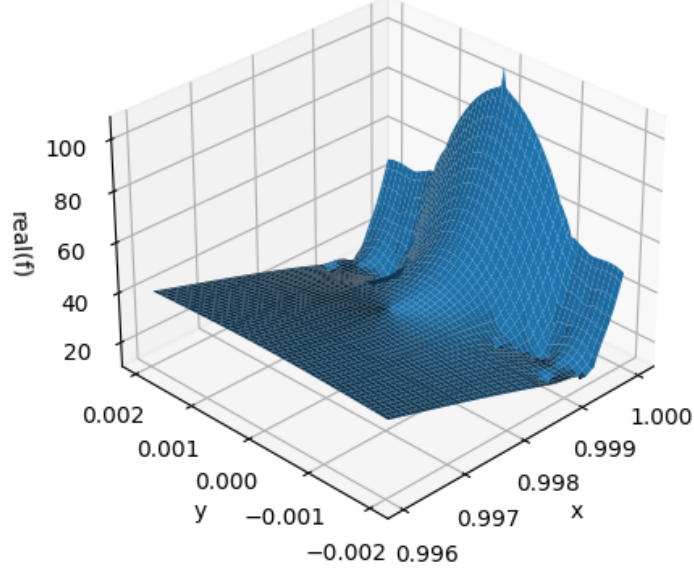


Figure 1: Saddle-point of $\Re(f(z))$

approximation.

First, the contour Γ is split into two parts $\Gamma = \Gamma_0 \cup \Gamma_1$, where:

- $\Gamma_0 = \{z_0 e^{i\theta} \text{ with } -\theta_0(z_0) \leq \theta \leq \theta_0(z_0)\}$
- $\Gamma_1 = \{z_0 e^{i\theta} \text{ with } \theta_0(z_0) \leq \theta \leq 2\pi - \theta_0(z_0)\}$

and $\theta_0(z_0)$ has to be defined accordingly.

The following three conditions must be fulfilled [FS09]:

- C1: Tail pruning. On the contour Γ_1 , the tails integral is negligible:

$$\oint_{\Gamma_1} \frac{G(z)}{z^{W+1}} dz = o\left(\oint_{\Gamma} \frac{G(z)}{z^{W+1}} dz\right)$$

- C2: Central approximation. Along Γ_0 , a quadratic expansion is valid for $f(z) = \log(G(z)) - W \log(z)$:

$$f(z_0 e^{i\theta}) \sim f(z_0) - \frac{\alpha_2(z_0)}{2} \theta^2 + o(\theta^3)$$

where $f'(z_0) = 0$ (or equivalently $z_0 \frac{G'(z_0)}{G(z_0)} = W + 1$) and $\alpha_2(z_0) = z_0^2 f''(z_0)$.

- C3: Tail completion. The incomplete Gaussian integral resulting from C2, taken over Γ_0 , is asymptotically equivalent to a complete Gaussian integral:

$$\int_{-\theta_0}^{+\theta_0} e^{-\alpha_2 \theta^2 / 2} d\theta \sim \frac{1}{\sqrt{|\alpha_2|}} \int_{-\infty}^{+\infty} e^{-t^2 / 2} dt = \sqrt{\frac{2\pi}{|\alpha_2|}}$$

this assumes that $|\alpha_2 \theta_0^2| \gg 1$.

The saddle-point approximation is defined by the following theorem.

Theorem 3. *Assuming that conditions C1, C2, and C3 are verified, we have (Chapter VIII of [FS09]):*

$$\frac{1}{2\pi i} \oint_{\Gamma} \frac{G(z)}{z^{W+1}} dz \sim \frac{G(z_0)}{z_0^W \sqrt{2\pi |f''(z_0)|}} \quad (5)$$

where $\sim$ is the asymptotic equivalence for large W .

4.2 Application to key ranking

The contour integral defining the key rank in Eq. 3 leads to the definition of $G(z)$ and $f(z)$:

$$\begin{aligned} G(z) &= \frac{1}{1-z} \prod_{k=0}^{K-1} \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right) \\ f(z) &= -W \log(z) - \log(1-z) + \sum_{k=0}^{K-1} \log \left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right) \end{aligned} \quad (6)$$

The saddle point z_0 is the solution $f'(z_0) = 0$ where $f'(z)$ is defined by Eq. 7. As stated above, z_0 exists, is unique, real and $0 < z_0 < 1$.

$$f'(z) = -\frac{W}{z} + \frac{1}{1-z} + \sum_{k=0}^{K-1} \left(\frac{\sum_{j=0}^{N-1} w_{k,j} z^{w_{k,j}-1}}{\sum_{j=0}^{N-1} z^{w_{k,j}}} \right) \quad (7)$$

To formally prove that conditions C1, C2, and C3 are satisfied, one must derive the analytic formula of z_0 as a function of W and the weights $w_{k,j}$. Unfortunately, the complexity of the function $f(z)$ makes this task impossible. This would be equivalent to deriving an analytic expression for the real root of a polynomial of very large degree. We must rely on simulation results to show that these three conditions can be satisfied. To do so, we reuse a weight matrix defined in the source code of the Labyntyr project (<https://github.com/sca-research/labyntyr>) for an AES application ($K = 16$) and provided as an artifact. The key point is how to set the value of θ_0 , which defines the two contours Γ_0 and Γ_1 . The value of θ_0 depends on the parameter W . Consequently, we run simulations with W ranging from small to large values : $W \in \{5, 10, 50, 100, 500\}$. The rank computed for each value of W is given in Table 2. It is given in $\log_2$ scale and covers small to large rank values.

Table 2: Key rank as a function of W .

W	5	10	50	100	500
R	8.3	13.3	35.4	51.7	104.5

First, we define $R_{\Gamma} = \oint_{\Gamma} \frac{G(z)}{z^{W+1}} dz$ and similarly R_{Γ_0} and R_{Γ_1} for contours Γ_0 and Γ_1 defined in Section 4.1. We evaluate these values using a numerical integration method. The variable R_{Γ_1}/R_{Γ} is computed to check the validity of condition C1. Concerning condition C2, the following relative error is defined:

$$E_1(\theta_0) = \left| \frac{f(z_0 e^{i\theta_0}) - f(z_0) + \frac{\alpha_2(z_0)}{2} \theta_0^2}{f(z_0 e^{i\theta_0})} \right|$$

Similarly, the following error measure is defined for C3:

$$E_2(\theta) = \left| \frac{R_{\Gamma_0} - R_G}{R_G} \right|$$

where $R_G = \sqrt{\frac{2\pi}{\alpha_2(z_0)}}$. It measures the error of approximating R_{Γ_0} with the Gaussian integral. Conditions C1, C2 and C3 are satisfied if R_{Γ_1}/R_{Γ} , $E_1(\theta_0)$ and $E_2(\theta_0)$ are small. Figure 2 shows the evolution of the ratio R_{Γ_1}/R_{Γ} as a function of θ_0 for the various values of W . Figure 3 and Figure 4 plot $E_1(\theta_0)$ and $E_2(\theta_0)$. We observe the three conditions are easily verified for large W . They are simultaneously satisfied when θ_0 is localized at the first local minimum of the curve R_{Γ_1}/R_{Γ} (Figure 2). The corresponding values are given in Table 3.

These simulations show that it is possible to find an appropriate value of θ_0 such that the conditions that must be satisfied to use the saddle-point approximation can be met. This is even true when W is not large (e.g. $W = 5$). Several simulation results not presented in this paper come to the same conclusion. Thus, even if a theoretical proof is still to be found, we strongly believe that the saddle-point approximation of key rank given by Eq. 5 is valid.

Table 3: θ_0 as a function of W .

W	5	10	50	100	500
θ_0 (rad)	0.52	0.34	0.13	0.15	0.05

In order to ease the readability of the measure, the key rank is converted to its $\log_2$ value:

$$R_{SP} = \frac{f(z_0)}{\log(2)} - \frac{1}{2} \log_2 \left(2\pi |f''(z_0)| \right) \quad (8)$$

where $f(z)$ is defined by Eq. 6 and the expression of $f''(z)$ is:

$$f''(z) = \frac{W}{z^2} + \frac{1}{(1-z)^2} + \frac{1}{z^2} \sum_{k=0}^{K-1} \left(\frac{\left(\sum_{j=0}^{N-1} w_{k,j}^2 z^{w_{k,j}} \right) - \left(\sum_{j=0}^{N-1} w_{k,j} z^{w_{k,j}} \right)^2}{\left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right)^2} - \frac{\left(\sum_{j=0}^{N-1} w_{k,j} z^{w_{k,j}} \right)^2}{\left(\sum_{j=0}^{N-1} z^{w_{k,j}} \right)^2} \right) \quad (9)$$

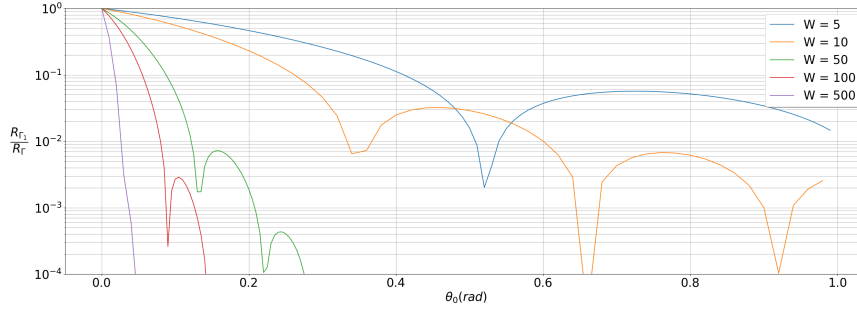


Figure 2: $\frac{R_{\Gamma_1}}{R_{\Gamma}}$ as a function of θ_0 for $W \in \{5, 10, 50, 100, 500\}$

5 Validation with simulations

The saddle-point approximation given by Eq. 8 is compared with the key rank estimation given by the original KP algorithm [MOOS15] (denoted 'KP-LP') and the 'batch' histogram

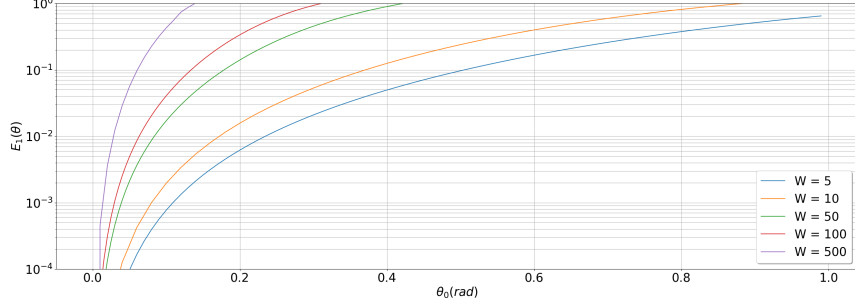


Figure 3: Approximation error on $f(z_0 e^{i\theta_0})$ as a function of θ_0 for $W \in \{5, 10, 50, 100, 500\}$

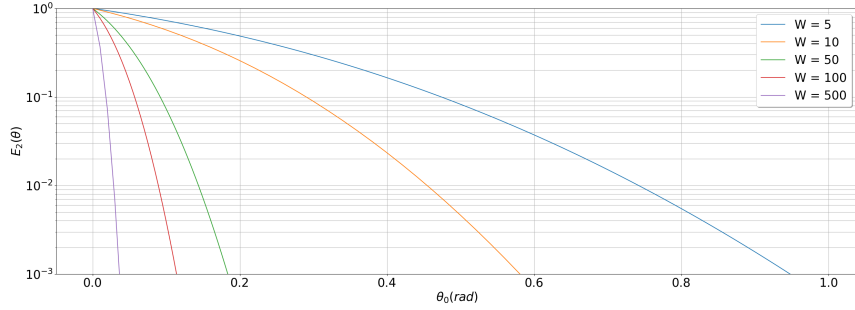


Figure 4: Approximation error on R_{Γ_0} as a function of θ_0 for $W \in \{5, 10, 50, 100, 500\}$

method [Gro18]. The latter is suitable for large key sizes (e.g. RSA or Dilithium) and has a fast execution time.

5.1 Simulation setup

All the key ranking estimation methods are implemented in Python and the source code will be available in the open source library SecBench (<https://github.com/CEA-Leti/secbench>). To support very large integer multiplications, histogram convolution and the original KP algorithm are optionally implemented with a library based on the Schönhage–Strassen algorithm (<https://github.com/rtitiu/polymul-approx-ffts/tree/main>). The calculation of z_0 is implemented with the Scipy function `root_scalar`. As a result, we do not compare the execution time of optimized implementations.

The weights $w_{k,j}$ are generated with K template attacks targeting the output of the AES Sbox. The leakage is modeled by $L_{k,n} = HW(SBOX(x_{k,n} \oplus s_k)) + b_{k,n}$ where $HW(x)$ is the Hamming weight of x . The variable $x_{k,n}$ is the k^{th} plaintext byte used for the n^{th} trace, s_k is the k^{th} key byte and $b_{k,n}$ is a zero mean Gaussian random variable with variance σ^2 . The probability associated with the value j of the k^{th} key byte, after a maximum likelihood estimation with $L = 100$ traces is:

$$p_{k,j} = \lambda \exp \left(- \frac{\sum_{n=0}^{L-1} |L_{k,n} - HW(SBOX(j \oplus x_{k,n}))|^2}{2\sigma^2} \right)$$

where λ is a normalization factor such that $\sum_{j=0}^{N-1} p_{k,j} = 1$.

The floating point weights $-\log(p_{k,j})$ are then mapped to $p = 9$ bits integers $w_{k,j}$ according

to Algorithm 1. The number of bins of the histogram method is thus set to 2^9 .

5.2 Comparison of rank estimation methods

Fig. 5 plots the mean rank error between the saddle-point (SP) approximation and the 'KP-LP' estimation as a function of the noise variance. Fig. 6 does the same between the 'batch' and 'KP-LP' methods. We observe a small error for the SP approximation which decreases as σ^2 or K increase. This is due to the increase of W with σ^2 and K and an approximation error proportional to $1/W$ [Olv74]. On the other hand, the error is large for the 'batch' estimation method.

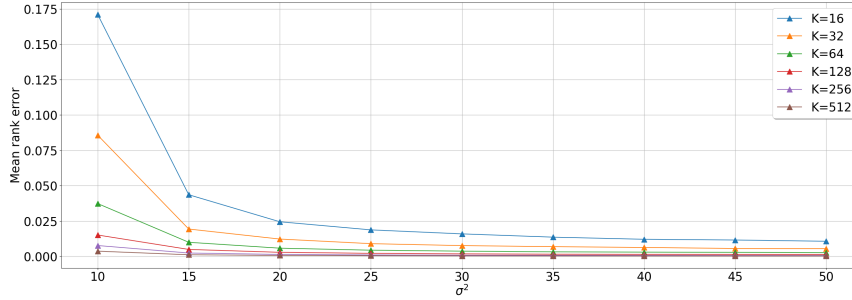


Figure 5: Rank estimation error of the saddle-point approximation for large K .

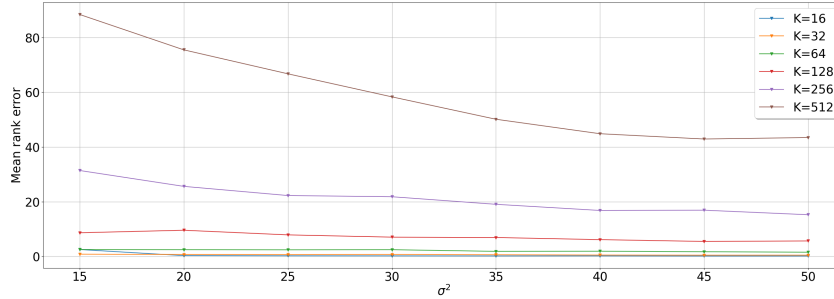


Figure 6: Rank estimation error of the batch histogram method for large K .

Fig. 7 shows the average execution time of the SP approximation, the 'KP-LP' and the 'batch' algorithms as a function of noise variance for various key sizes (K bytes). Up to $K = 128$, SP approximation and 'KP-LP' are equivalent. The 'batch' algorithm is a little bit slower. For larger key sizes ($K \in \{128, 256, 512\}$), the 'KP-LP' algorithm must be implemented with a dedicated library to support very large integers. This increases significantly the mean execution time, as shown in Figure 8. The 'KP-LP' algorithm is thus slower than the SP and 'batch' algorithm with this Python implementation.

Fig. 9 focuses on the SP approximation and the 'batch' algorithm for large K sizes. For a fixed value of K , the curves are almost flat, meaning that the execution time is independent of the noise variance. The SP approximation is almost three times faster than the 'batch' algorithm $K = 512$. Fig. 10 plots the mean execution time measured with the SP approximation as a function of the key size K (in, bytes) for three noise variances. It is

almost independent of the noise variance and linear with K . Thus, the SP approximation is a solution that is both fast, accurate and its mean execution time is linear with the key size. Moreover, it does not need to use a dedicated library to handle large integers making its implementation straightforward for non-specialists.

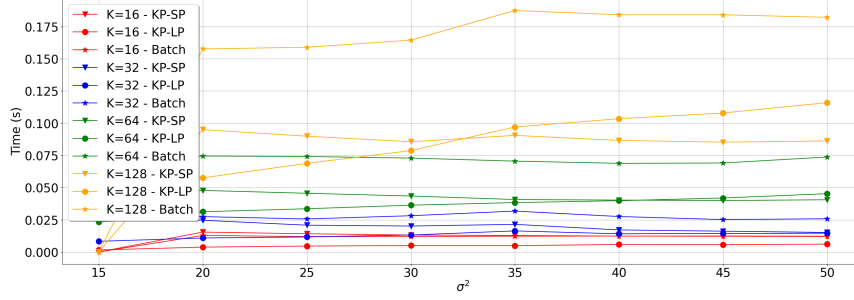


Figure 7: Mean execution time for $K \in \{16, 32, 64, 128\}$: SP ($\blacktriangledown$), 'KP-LP' ($\bullet$), 'batch' ($\star$)

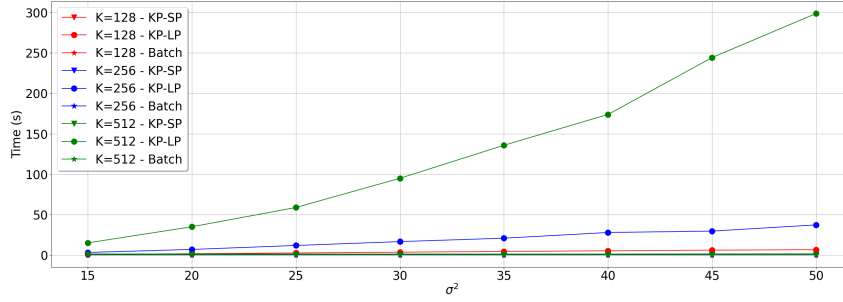


Figure 8: Mean execution time for $K \in \{128, 256, 512\}$: SP ($\blacktriangledown$), 'KP-LP' ($\bullet$), 'batch' ($\star$)

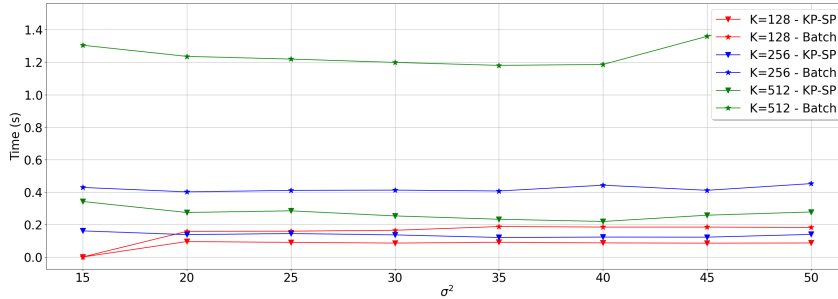


Figure 9: Mean execution time for $K \in \{128, 256, 512\}$: SP ($\blacktriangledown$) and 'batch' ($\star$)

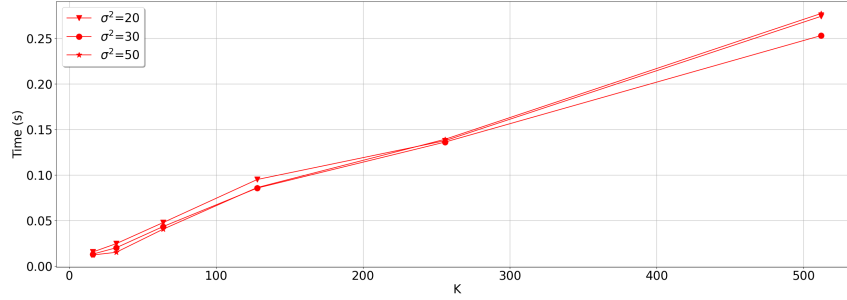


Figure 10: Mean execution time as a function of K for the SP approximation. We observe a linear dependency with K and an independence with σ^2 .

6 Conclusion

Key ranking is a critical issue for evaluators since it measures the remaining complexity after a side-channel attack without the need to implement the key enumeration algorithm. An analytic expression of the key rank based on the Knapsack Problem has been derived. It is inspired by a recent article in statistical physics. This formula is used to prove the mathematical equivalence with the histogram method and the PRO algorithm. A saddle-point approximation of the key rank is also derived based on this analytic expression. The approximation is given by a low complexity solution of a mathematical problem. Simulation results show that the approximation is very tight and the estimation error decreases as the key rank increases. The execution time of the approximation is linear in the key size, and results based on a non-optimized Python implementation show that it is both fast and accurate. This makes it a good solution for scaling to very large keys.

References

- [BLv15] Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. Cryptology ePrint Archive, Report 2015/221, 2015. <https://eprint.iacr.org/2015/221>.
- [CDS23] Giovanni Camurati, Matteo Dell’Amico, and François-Xavier Standaert. Merank: Monte carlo key rank estimation for side-channel security evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(1):277–300, 2023. doi:10.3929/ethz-b-000585426.
- [CP17] Marios O. Choudary and P. G. Popescu. Back to Massey: Impressively fast, scalable and tight security evaluation tools. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 367–386. Springer, Heidelberg, September 2017. doi:10.1007/978-3-319-66787-4_18.
- [DW22] Liron David and Avishai Wool. Rank estimation with bounded error via exponential sampling. *Journal of Cryptographic Engineering*, 12(2):151–168, 2022. doi:10.1007/s13389-021-00269-4.
- [FS09] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. doi:10.1017/CB09780511801655.

- [GGP⁺15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, Heidelberg, March 2015. doi:[10.1007/978-3-662-48116-5_6](https://doi.org/10.1007/978-3-662-48116-5_6).
- [Gro18] Vincent Grosso. Scalable key rank estimation (and key enumeration) algorithm for large keys. In *International Conference on Smart Card Research and Advanced Applications*, pages 80–94. Springer, 2018. doi:[10.1007/978-3-030-15462-2_6](https://doi.org/10.1007/978-3-030-15462-2_6).
- [HO24] Rebecca Hay and Elisabeth Oswald. Key rank estimation methods: Comparisons and practical considerations. *IACR Communications in Cryptology*, 1(3), 2024. doi:[10.62056/aytxl86bm](https://doi.org/10.62056/aytxl86bm).
- [KPP⁺04] Hans Kellerer, Ulrich Pferschy, David Pisinger, Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Multidimensional knapsack problems*. Springer, 2004. doi:[10.1007/978-3-540-24777-7_9](https://doi.org/10.1007/978-3-540-24777-7_9).
- [MM09] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009. doi:[10.1093/acprof:oso/9780198570837.001.0001](https://doi.org/10.1093/acprof:oso/9780198570837.001.0001).
- [MMO18] Daniel P. Martin, Luke Mather, and Elisabeth Oswald. Two sides of the same coin: Counting and enumerating keys post side-channel attacks revisited. In Nigel P. Smart, editor, *Topics in Cryptology – CT-RSA 2018*, volume 10808 of *Lecture Notes in Computer Science*, pages 394–412. Springer, Heidelberg, April 2018. doi:[10.1007/978-3-319-76953-0_21](https://doi.org/10.1007/978-3-319-76953-0_21).
- [MMOS16] Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 548–572. Springer, Heidelberg, December 2016. doi:[10.1007/978-3-662-53887-6_20](https://doi.org/10.1007/978-3-662-53887-6_20).
- [MOOS15] Daniel P. Martin, Jonathan F. O’Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, Heidelberg, November / December 2015. doi:[10.1007/978-3-662-48800-3_13](https://doi.org/10.1007/978-3-662-48800-3_13).
- [Olv74] Frank Olver. *Introduction to Asymptotics and Special Functions*. Academic Press, 1974. doi:[10.1137/1.9781611973648](https://doi.org/10.1137/1.9781611973648).
- [PGS15] Romain Pousier, Vincent Grosso, and François-Xavier Standaert. Comparing approaches to rank estimation for side-channel security evaluations. In *International Conference on Smart Card Research and Advanced Applications, CARDIS*, pages 125–142. Springer, 2015. doi:[10.1007/978-3-319-31271-2_8](https://doi.org/10.1007/978-3-319-31271-2_8).
- [RPC22] Anca Rădulescu, Pantelimon G Popescu, and Marios O Choudary. Ge vs gm: Efficient side-channel security evaluations on full cryptographic keys. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 886–905, 2022. doi:[10.46586/tches.v2022.i4.886-905](https://doi.org/10.46586/tches.v2022.i4.886-905).

- [SDK75] Harvey M Salkin and Cornelis A De Kluyver. The knapsack problem: a survey. *Naval Research Logistics Quarterly*, 22(1):127–144, 1975. doi:[10.1002/nav.3800220110](https://doi.org/10.1002/nav.3800220110).
- [VGRS13] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, Heidelberg, August 2013. doi:[10.1007/978-3-642-35999-6_25](https://doi.org/10.1007/978-3-642-35999-6_25).
- [Wil24] Mobolaji Williams. Large-W limit of the knapsack problem. *Physical Review E*, 109(4):044151, 2024. doi:[10.1103/PhysRevE.109.044151](https://doi.org/10.1103/PhysRevE.109.044151).
- [YMO22] Rebecca Young, Luke Mather, and Elisabeth Oswald. Comparing key rank estimation methods. In *International Conference on Smart Card Research and Advanced Applications*, pages 188–204. Springer, 2022.
- [ZDF20] Ziyue Zhang, A. Adam Ding, and Yunsi Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):26–48, 2020. <https://tches.iacr.org/index.php/TCHES/article/view/8543>. doi:[10.13154/tches.v2020.i2.26-48](https://doi.org/10.13154/tches.v2020.i2.26-48).